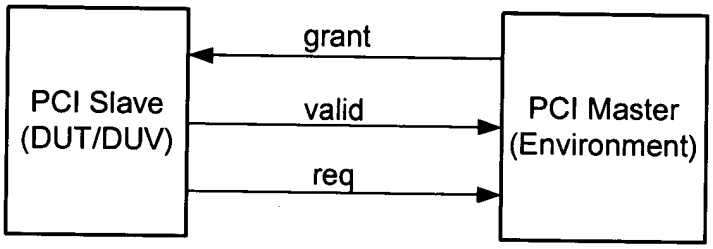

Figure 1

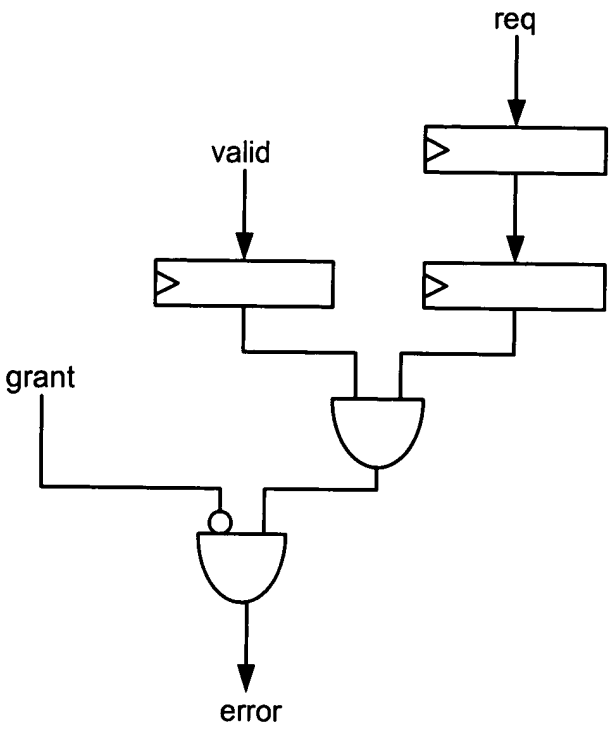

Figure 2

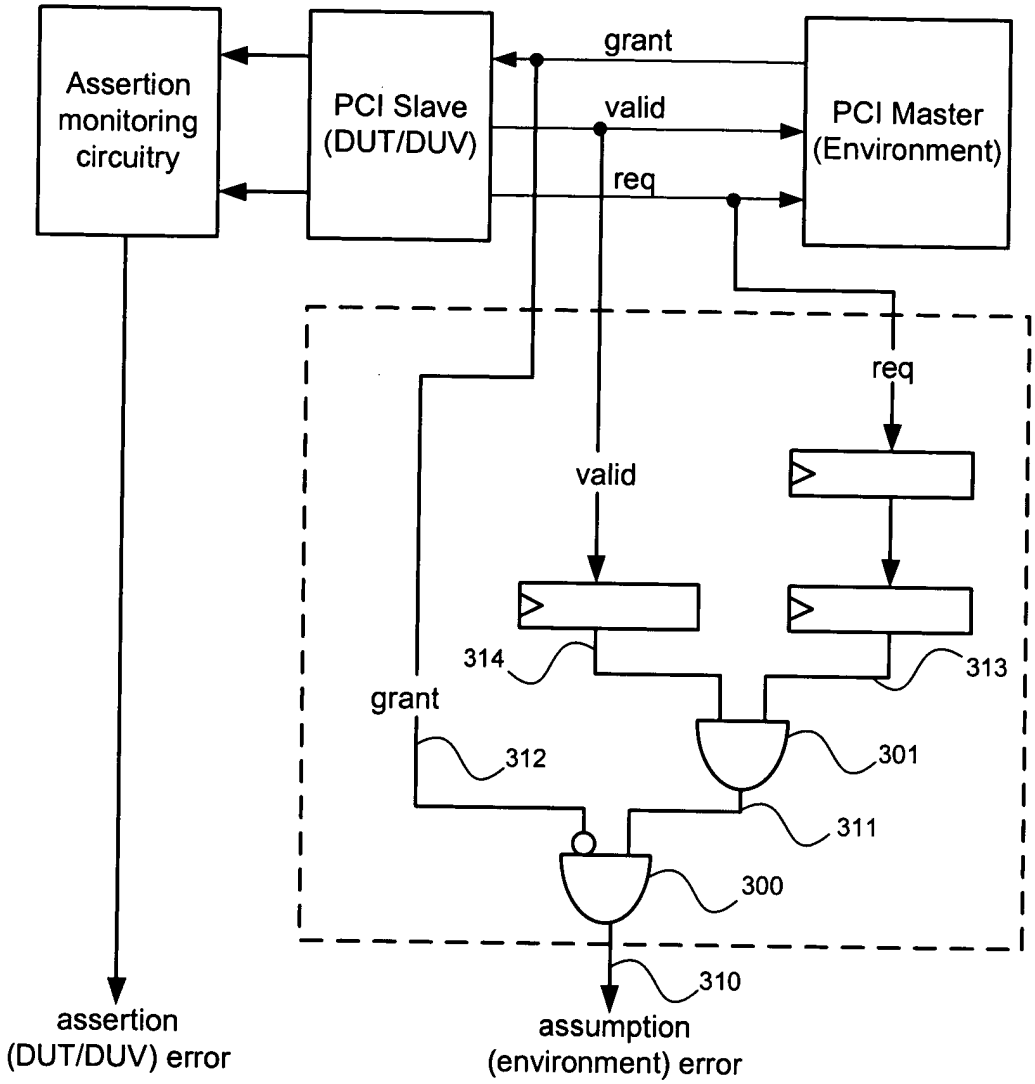
Assertion monitoring circuitry
PCI Slave (DUT/DUV)
PCI Master (Environment)
grant
valid
req
req
valid
314
313
grant
312
301
311
300
310
assertion (DUT/DUV) error
assumption (environment) error

Figure 3

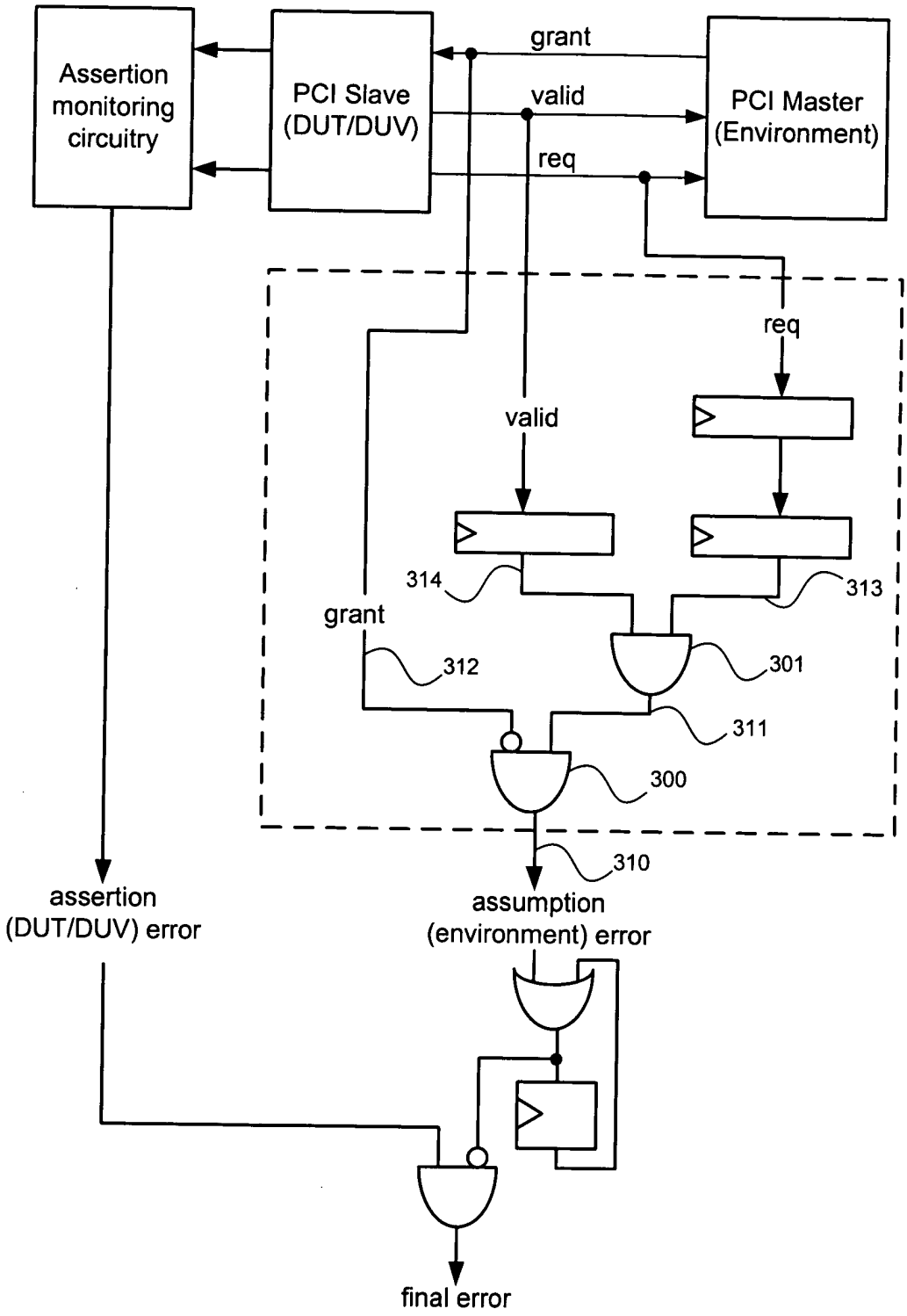
Assertion monitoring circuitry
PCI Slave (DUT/DUV)
PCI Master (Environment)
grant
valid
req
req
valid
314
313
grant
312
301
311
300
310
assertion (DUT/DUV) error
assumption (environment) error
final error

Figure 4

x, y, z, as DUT/DUV
inputs, are RV's

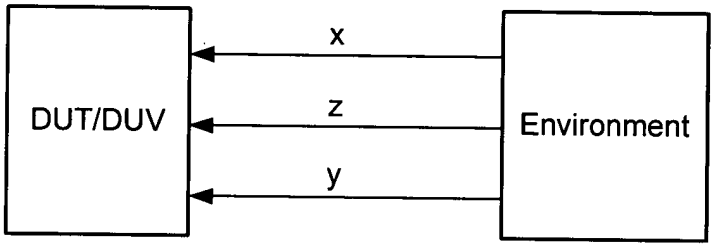

Figure 5

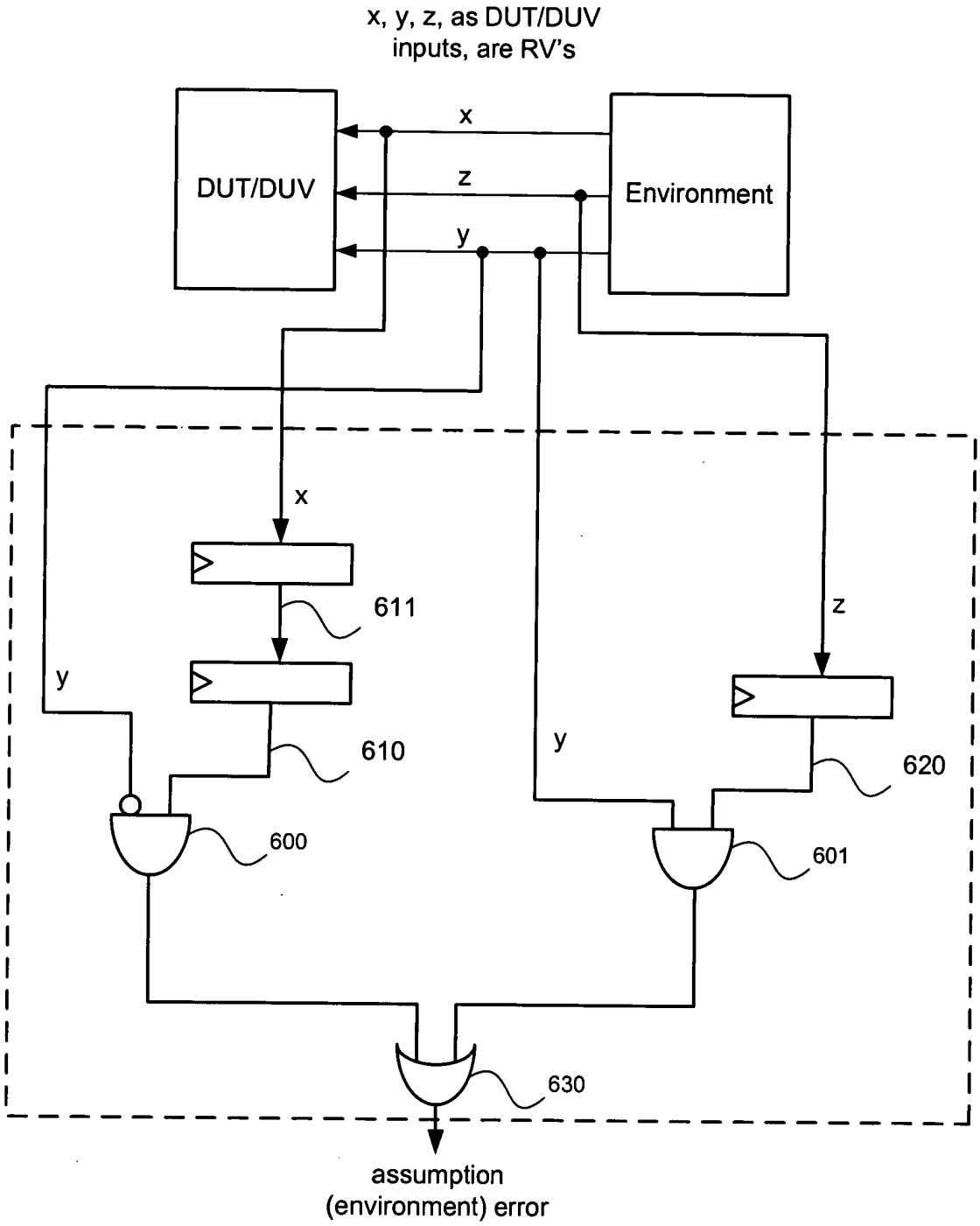
x, y, z, as DUT/DUV
inputs, are RV's
DUT/DUV
Environment
x
z
y
x
y
z
y
611
610
620
600
601
630
assumption
(environment) error

Figure 6

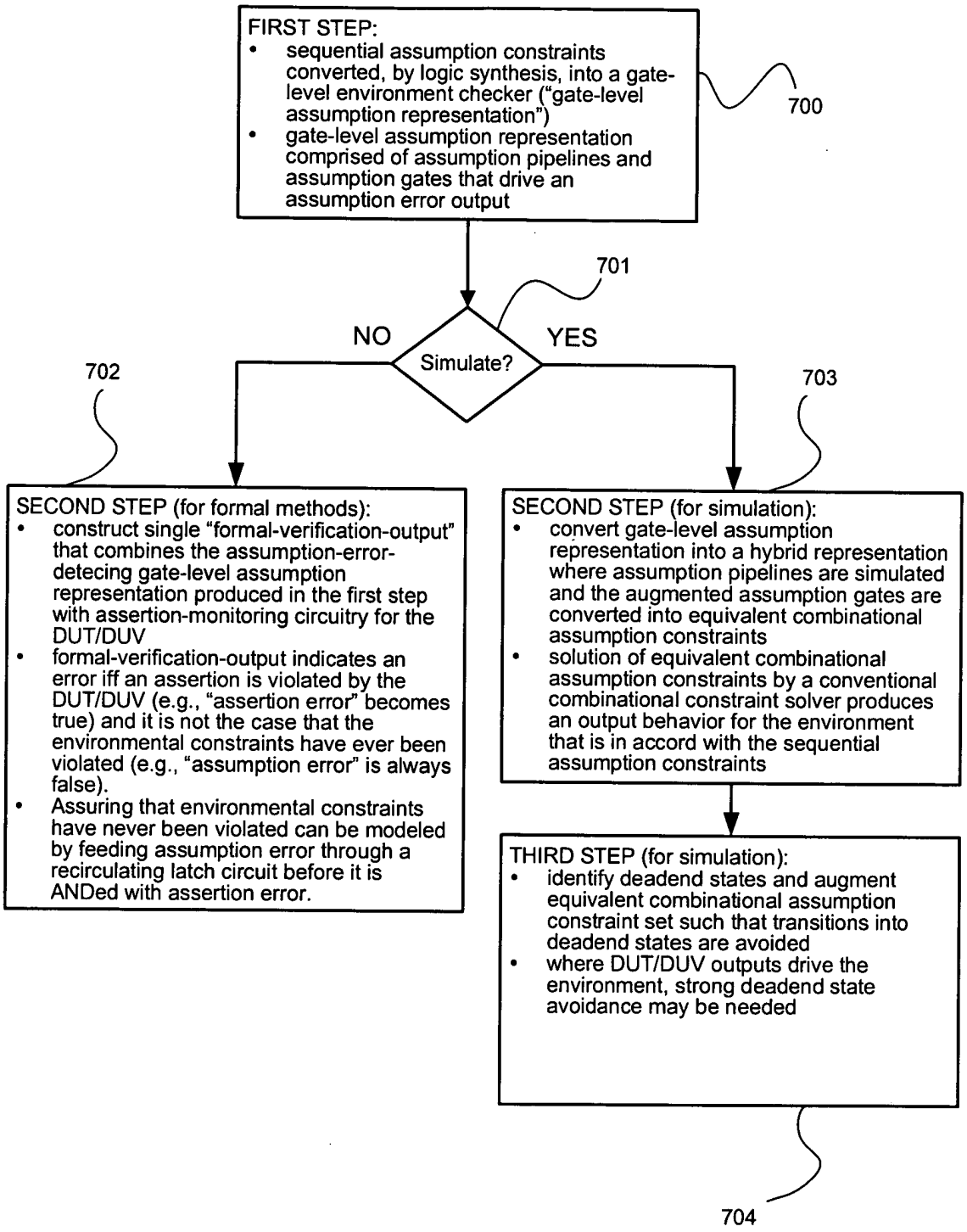
FIRST STEP:
• sequential assumption constraints converted, by logic synthesis, into a gate-level environment checker ("gate-level assumption representation")
• gate-level assumption representation comprised of assumption pipelines and assumption gates that drive an assumption error output
700
701
Simulate?
NO
YES
702
703
SECOND STEP (for formal methods):
• construct single "formal-verification-output" that combines the assumption-error-detecing gate-level assumption representation produced in the first step with assertion-monitoring circuitry for the DUT/DUV
• formal-verification-output indicates an error iff an assertion is violated by the DUT/DUV (e.g., "assertion error" becomes true) and it is not the case that the environmental constraints have ever been violated (e.g., "assumption error" is always false).
• Assuring that environmental constraints have never been violated can be modeled by feeding assumption error through a recirculating latch circuit before it is ANDed with assertion error.
SECOND STEP (for simulation):
• convert gate-level assumption representation into a hybrid representation where assumption pipelines are simulated and the augmented assumption gates are converted into equivalent combinational assumption constraints
• solution of equivalent combinational assumption constraints by a conventional combinational constraint solver produces an output behavior for the environment that is in accord with the sequential assumption constraints
THIRD STEP (for simulation):
• identify deadend states and augment equivalent combinational assumption constraint set such that transitions into deadend states are avoided
• where DUT/DUV outputs drive the environment, strong deadend state avoidance may be needed
704

FIGURE 7A

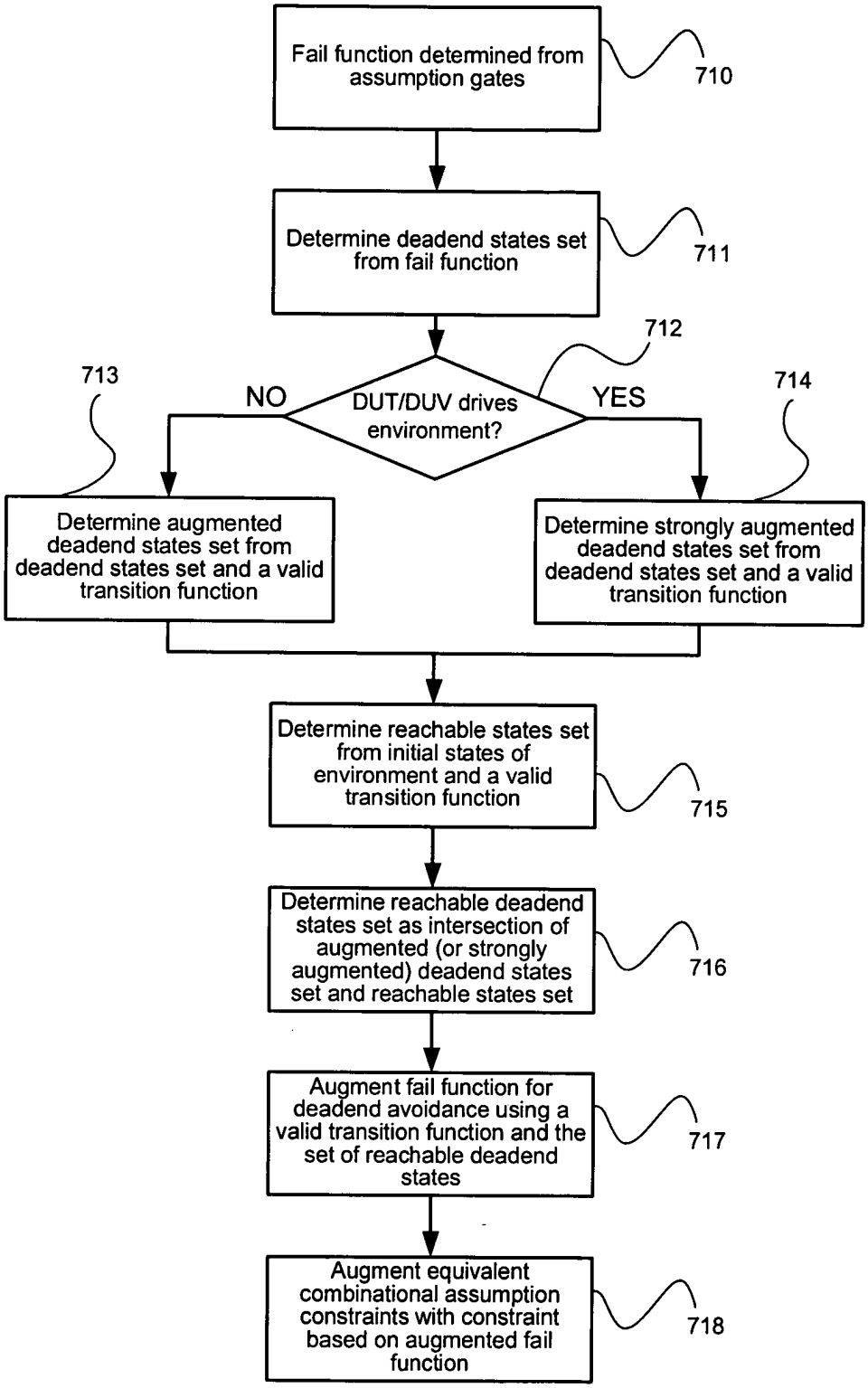
Fail function determined from assumption gates
710
Determine deadend states set from fail function
711
712
DUT/DUV drives environment?
NO
YES
713
714
Determine augmented deadend states set from deadend states set and a valid transition function
Determine strongly augmented deadend states set from deadend states set and a valid transition function
Determine reachable states set from initial states of environment and a valid transition function
715
Determine reachable deadend states set as intersection of augmented (or strongly augmented) deadend states set and reachable states set
716
Augment fail function for deadend avoidance using a valid transition function and the set of reachable deadend states
717
Augment equivalent combinational assumption constraints with constraint based on augmented fail function
718

FIGURE 7B

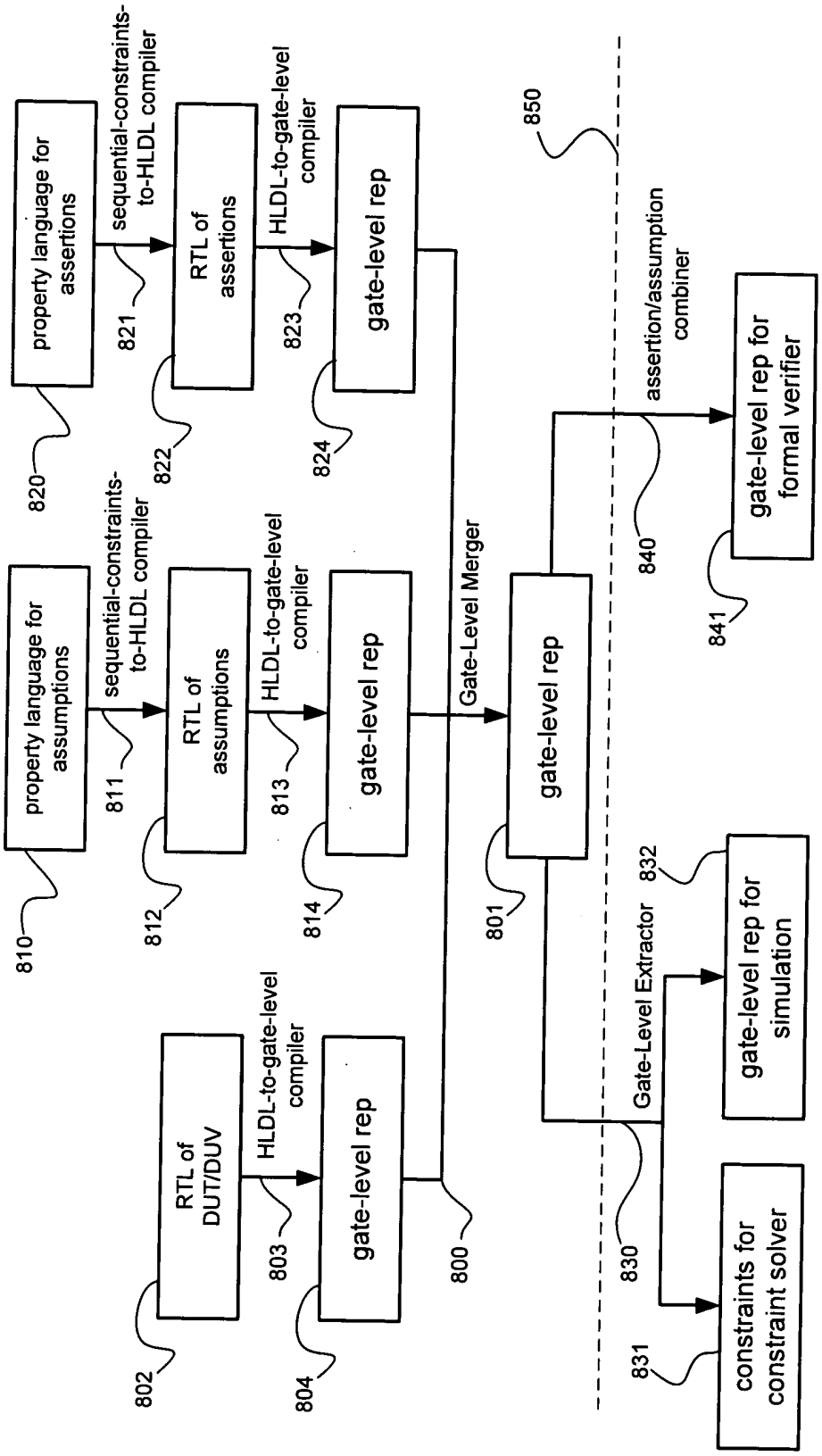
property language for assertions
sequential-constraints-to-HLDL compiler
RTL of assertions
HLDL-to-gate-level compiler
gate-level rep
property language for assumptions
sequential-constraints-to-HLDL compiler
RTL of assumptions
HLDL-to-gate-level compiler
gate-level rep
RTL of DUT/DUV
HLDL-to-gate-level compiler
gate-level rep
Gate-Level Merger
gate-level rep
850
assertion/assumption combiner
gate-level rep for formal verifier
Gate-Level Extractor
gate-level rep for simulation
constraints for constraint solver
810
811
812
813
814
820
821
822
823
824
800
801
802
803
804
830
831
832
840
841

FIGURE 8

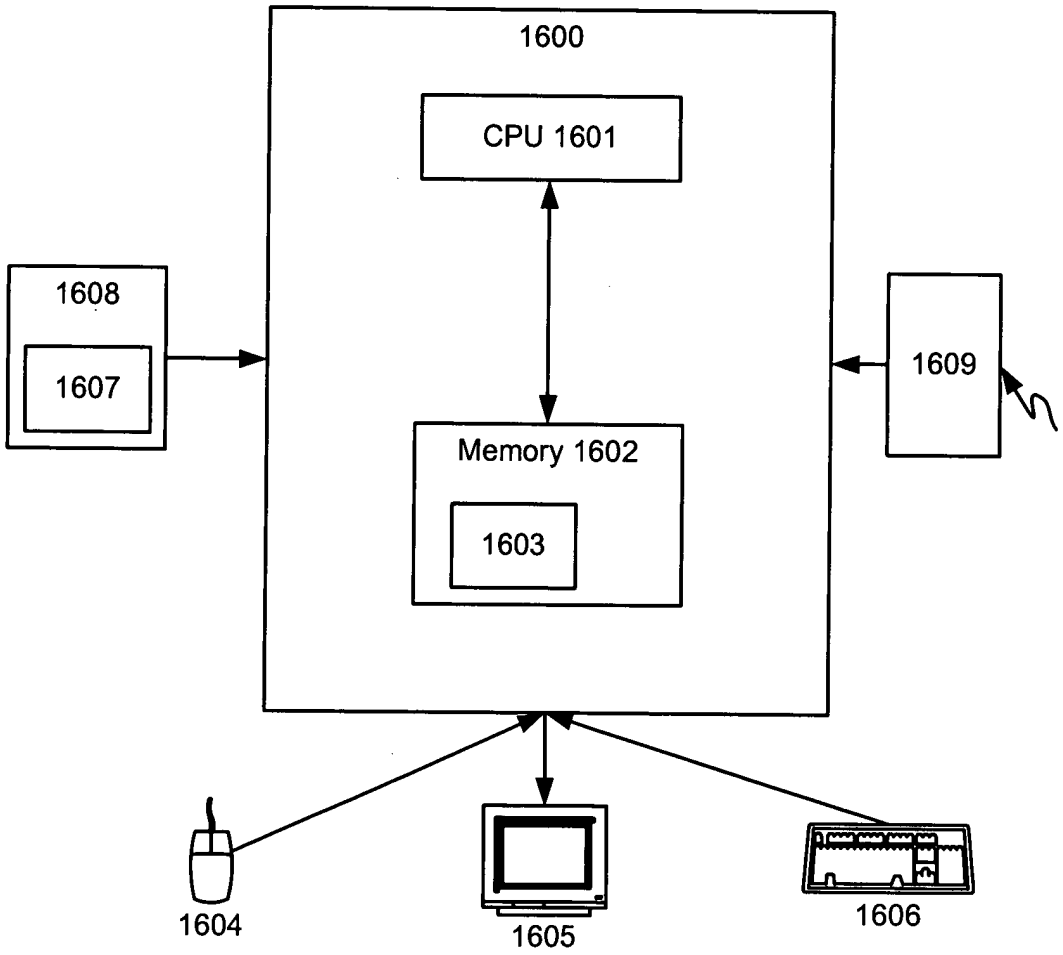

FIGURE 9